



# UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE  
United States Patent and Trademark Office  
Address: COMMISSIONER FOR PATENTS  
P.O. Box 1450  
Alexandria, Virginia 22313-1450  
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/911,819	07/24/2001	John T. Micco	04899-046001	6291

7590 03/25/2005

Kevin J. Canning, Esq.  
Lahive & Cockfield, LL.P  
28 State Street  
Boston, MA 02109

EXAMINER

VU, TUAN A

ART UNIT	PAPER NUMBER
----------	--------------

2193

DATE MAILED: 03/25/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

<b>Office Action Summary</b>	Application No. 09/911,819	Applicant(s) MICCO ET AL.	
	Examiner Tuan A Vu	Art Unit 2124	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

#### Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

#### Status

- 1) ☒ Responsive to communication(s) filed on 11/03/2004.
- 2a) ☒ This action is **FINAL**.                      2b) ☐ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

#### Disposition of Claims

- 4) ☒ Claim(s) 1-56 is/are pending in the application.
- 4a) Of the above claim(s) \_\_\_\_\_ is/are withdrawn from consideration.
- 5) ☐ Claim(s) \_\_\_\_\_ is/are allowed.
- 6) ☒ Claim(s) 1-56 is/are rejected.
- 7) ☐ Claim(s) \_\_\_\_\_ is/are objected to.
- 8) ☐ Claim(s) \_\_\_\_\_ are subject to restriction and/or election requirement.

#### Application Papers

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☐ The drawing(s) filed on \_\_\_\_\_ is/are: a) ☐ accepted or b) ☐ objected to by the Examiner.  
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).  
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

#### Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All    b) ☐ Some \*    c) ☐ None of:
1. ☐ Certified copies of the priority documents have been received.
2. ☐ Certified copies of the priority documents have been received in Application No. \_\_\_\_\_.
3. ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).
- \* See the attached detailed Office action for a list of the certified copies not received.

#### Attachment(s)

- |  |   |
|--|---|
| 1) <input checked="" type="checkbox"/> Notice of References Cited (PTO-892)  | 4) <input type="checkbox"/> Interview Summary (PTO-413)<br>Paper No(s)/Mail Date. _____ |
| 2) <input type="checkbox"/> Notice of Draftsperson's Patent Drawing Review (PTO-948)                                   | 5) <input type="checkbox"/> Notice of Informal Patent Application (PTO-152)             |
| 3) <input type="checkbox"/> Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08)<br>Paper No(s)/Mail Date _____ | 6) <input type="checkbox"/> Other: _____  |

### DETAILED ACTION

1. This action is responsive to the Applicant's response filed 11/03/2004.

As indicated in Applicant's response, claims 1-56 are pending in the office action.

#### *Double Patenting*

2. The nonstatutory double patenting rejection is based on a judicially created doctrine grounded in public policy (a policy reflected in the statute) so as to prevent the unjustified or improper timewise extension of the "right to exclude" granted by a patent and to prevent possible harassment by multiple assignees. See *In re Goodman*, 11 F.3d 1046, 29 USPQ2d 2010 (Fed. Cir. 1993); *In re Longi*, 759 F.2d 887, 225 USPQ 645 (Fed. Cir. 1985); *In re Van Ornum*, 686 F.2d 937, 214 USPQ 761 (CCPA 1982); *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970); and, *In re Thorington*, 418 F.2d 528, 163 USPQ 644 (CCPA 1969).

A timely filed terminal disclaimer in compliance with 37 CFR 1.321(c) may be used to overcome an actual or provisional rejection based on a nonstatutory double patenting ground provided the conflicting application or patent is shown to be commonly owned with this application. See 37 CFR 1.130(b).

Effective January 1, 1994, a registered attorney or agent of record may sign a terminal disclaimer. A terminal disclaimer signed by the assignee must fully comply with 37 CFR 3.73(b).

3. Claims 1, 11, 29, and 39 are provisionally rejected under the judicially created doctrine of obviousness-type double patenting as being unpatentable over claim 16 of copending Application No. 10/190288 (hereinafter '288). Although the conflicting claims are not identical, they are not patentably distinct from each other because of the following conflicts.

**As per claims 1 and 29, '288 claim 16 also recites**

---

processing a definition of a function associated with a first language (e.g. *source code for ... more functions in a first programming environment; ...processing the source code to create a component; component comprises... COM object; COM source code files include an Interface Description Language ... class definition and implementation files ...;process IDL files to produce...* - Note: class definition and implementation files in conjunction with *receiving source*

Art Unit: 2124

*code for one or more functions created in the first language read on definition of a function ... first language being processed ) to create description information about the function (Note: COM source file along with IDL and implementation files, export files, library file reads on description information about the function),*

the description being sufficient to enable translation of a call to the function into a call to a corresponding function in a second language (e.g. *component is usable by the application ... second programming environment to access ... functions of the component; converting the source code from a first language to a second programming language; compiling the converted source code files, converted COM source code files and processed IDL files to produce object files – Note: converted ... source code files; linking ... libraries support one or more functions ... object files to produce a version of the component read on translation of a call to the function into a call to a corresponding function; component ... usable by the ... program in a second programming environment reads on in a second language).*

But '288 does not recite that the translation using the description is being done 'without requiring processing the one or more functions of the component'. It was known in the art at the time of the application that a definition language file (e.g. IDL) is for setting the specification or language specific declaration for a function, class or method adapted to be used for interfacing in heterogeneous environment in the art of software development; and that is purported to obviate extraneous re-conversion/compiling resources by the target environment for having to process the function received from another environment. Besides, '288 further recites '*...IDL compile ... component type library file ... wherein processing ... version of the component ... does not include type information'* ( see claim 15 or 16). Thus, this lends to the interpretation that by

Art Unit: 2124

compiling the IDL and associated libraries of the functions, it is not necessary to process the target programming language rules on how the functions are to be defined because the component library file from IDL compilation has provided the sufficient information ( IDL , as noted from above); hence, without need to verify of the type information when the functions is implemented in the second language. Thus, '288 has read on *translation without requiring processing the one or more functions of the component*, i.e. the object files produced from the IDL and related libraries and header files; or if not, the providing of IDL tool so to obviate reprocessing a function definition written in another language ( e.g. as required by instant claim 1) by a target heterogeneous environment would have been obvious based on the teachings of '288 as recited.

**As per claims 11 and 39**, '288 claim 16 also recites

file of description items (*an Interface Description Language ... class definition and implementation files ...* - Note: class definition and implementation files are equivalent to definition of a function being processed), description associated with a first language and sufficient to enable to translate a call to a corresponding call in a second language; and

using the file of description items to translate a first program call into a second program

~~file (component is usable by the application ... second programming environment to access ...~~

*functions of the component; converting the source code from a first language to a second programming language; compiling the converted source code files, converted COM source code files and processed IDL files to produce object files*); and further includes the implied recitation of 'without requiring processing the one or more functions of the component' as has been analyzed from above.

Art Unit: 2124

The claims from claims 1, 11, 29, and 39 are also rejected for being dependent on a rejected base claims.

This is a provisional obviousness-type double patenting rejection because the conflicting claims have not in fact been patented.

### ***Claim Rejections - 35 USC § 103***

4. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

5. Claims 1-20, and 29-48 are rejected under 35 U.S.C. 103(a) as being unpatentable over Shannon et al., "Mapping the Interface Description Language Type Model in C", November 1989, IEEE Transactions on Software Engineering, Vol. 15, No. 11, in view of Research Systems, "IDL", copyright 1994 (hereinafter Research\_Systems).

As per claim 1, Shannon discloses a method comprising  
creating description information or *IDL* about a function in a first language (e.g. ...  
*existing languages ... LISP, Diana structures ... mapped ... to C, Ada Breadboard Compiler -*  
Introduction pg. 1333-1334; *class, function* - Fig. 1 pg. 1335; *Process declaration* - Fig. 2; *IDL*  
*specifications* - Fig. 3, pg. 1336 – Note: The conversion based on a special package extending  
existing language structures to target C or Ada reads on creating a description interface on  
structures, e.g. a class definition, in first language, - into Ada function or C data structures; *data*  
*structures found in procedural programming languages* – 2<sup>nd</sup> para, right column, pg. 1334 –

Note: the use of existing programming constructs reads on function in first language when data structure is inherent to function of a language),

the description being sufficient to enable translation of a call to the function into a call to a corresponding function in a second language (*...should be permitted by the C compiler - Introduction pg. 1333-1334*) without requiring processing of the definition of the function (e.g. *... should be natural and not require knowledge of implementation details - pg. 1333, right column*).

But Shannon does not explicitly disclose processing a definition of a function associated with the first language to create the description information. In view of the suggested conversion from one language to another, e.g. from LISP structures to C as noted above, the processing of a source function ( i.e. class or data structures are implicit part of source function) in order to specify an interface definition language is strongly suggested. Research\_Systems, in a method using IDL analogous to Shannon's, discloses mapping to fourth-generation programming language like C, and using such IDL/programming language to implement applications as in Math functions, graph plotting, Image Processing ( pg. 1-5 – Note: all of these applications require functionalities that involve and require processing of input/output data), hence implies processing of each of those application function in order to create an intermediate mapping language, e.g. IDL. In other words, IDL entails a need to convey some functionalities expressed in one form into another form so that a first language leading to a second target language is heavily implied with IDL. In case Shannon does not already teach processing a application function in order to create an description file, it would have been obvious for one of ordinary skill in the art at the time the invention was made to apply the use of the IDL to implement

Art Unit: 2124

functions as taught by Research\_Systems because this interactive language would allow the developer to prototype and assist in developing complete real-world and highly data and processing extensive applications (math, image processing ) from one language, e.g. mathematical formulas or MathLab equations, to another language, e.g. C or Fortran, thus enabling advanced high-level programming language user-friendly constructs and compiling/debugging benefits that come with such language.

**As per claims 2 and 4**, Shannon discloses a file of description items and derived description information (e.g. sections II, III – pg. 1334-1339).

**As per claim 3**, Shannon ( combined with Research\_Systems) discloses using derived information about the function to translate the call to the function into a call to a corresponding function ( in a first language) in the second language (e.g. section III pg. 1336; Fig. 3 – Note: C is the second language and Math or graphical functions are in first language ). In view of the applications suggested by Research\_Systems from above, the examining of the function in those graphical or mathematical functions along with their definition would have been obvious based on the rationale of claim 1 because creating a IDL defining a target language constructs necessary implies the examining of how the source application function ( such as mentioned by Research\_Systems) is defined, i.e. an inherent-step-prior-to-specifying an interface language as intended by Shannon.

**As per claim 5**, Shannon discloses creation of C language constructs, hence has implicitly disclose the .lib files associated with the assembling of object files prior to linking in C. Therefore, Shannon has implicitly disclosed library wherein entries associated with the



Art Unit: 2124

assembling process in C compiler because at the time the invention was made it was a known concept that C compiler create lib files during a pre-linking compiler process.

**As per claims 6 and 7**, Shannon does not explicitly disclose processing of the function and deriving a number of declared formal inputs to the functions. But Shannon discloses a declaration with a set of input and output port (ch. B – pg. 1335; Fig. 2, pg. 1336 ) and input/output mapping ( pg. 1338, Private types - right column). Official notice is taken that the declaration of formal input and output, and scope of variables declared in a function in 4<sup>th</sup> generation like C was a known concept at the time the invention was made; hence the IDL input/output provision as suggested by Shannon should imply must-have analysis of the first language input/output formal parameters leading to creation of C formal parameters, i.e. list of input or output/return parameters otherwise the target function declaration would not result in a correct function declaration.

**As per claim 8**, Namespace and function scope involving local and global parameters are known in 4<sup>th</sup> generation like C as suggested from the above Official notice. Hence, the analysis of the first language function in order to derive a scope for declaring C function declaration would have obvious based on the implicit teachings as mentioned from the rationale of claims 6 and 7.

**As per claims 9 and 10**, Shannon ( in view of Research\_Systems) does not explicitly teach determining of variable arguments in a function and a variable return of results. Official notice is taken the advanced languages like C providing a variable number of arguments, e.g. input arglist[], or argv[]; or a variable output/return in form of an array, or pointer to a struct or linked list, was a known concept at the time the invention was made. Hence, determining

Art Unit: 2124

whether a function to have a variable input arguments or return variables would also have been obvious according to the rationale as used in claims 6-8 because complex and highly probabilistic applications as suggested by Research\_Systems, math or graphic processing, might entail a non-fixed amount of inputs or outputs, and analyzing such variability of parameters would allow the target code to accommodate for such eventuality, using the known approach provided by C as mentioned above.

**As per claim 11**, Shannon discloses providing a file of description items, information about a function in a first language (e.g. *Diana structures ... mapped ...to C, Ada Breadboard Compiler* – Introduction pg. 1333-1334), the description being sufficient to enable translation of a call to the function into a call to a corresponding function in a second language (...*should be permitted by the C compiler* - Introduction pg. 1333-1334) without requiring processing of the definition of the function (e.g. ... *should be natural and not require knowledge of implementation details* – pg. 1333, right column); and using the file description items to translate a function into a second program file (see Shannon: Introduction )

But Shannon fails to explicitly disclose using information about a function associated with the first language to translate a first program file into a second program file. But in view of the combined teachings of Shannon/Research\_Systems, a function in a first language, like Matlab or ADA functions file, the limitation of converting a first language file into a target file would have been obvious following the rationale as set forth in claim 1.

**As per claims 12-13**, referring to rationale of claims 6-7, the providing of descriptor for a declared a number of formal inputs or outputs to the functions would also have been obvious because analyzing the first language function necessarily requires implementing an interface

Art Unit: 2124

language with identification so to enable mapping such function data flow requirements, e.g. input or output number of parameters.

**As per claims 14-16**, refer to the rationale of claims 8-10.

**As per claim 17**, Shannon does not explicitly disclose for each call in the 1<sup>st</sup> program file, retrieving an item from the file of description items, and using information description in the item to translate the first language function into a call corresponding to the 2<sup>nd</sup> language; and storing the translated function in the second program file. But in view of teachings by the combination using Shannon IDL and function structures and type specifications in light of the application using IDL by Research\_Systems in claim 1, the above limitation is implicitly disclosed by Shannon, and would have been obvious according to the rationale of claim 1 and 11.

**As per claims 18-19**, refer to rationale as set forth in claims 15-16, respectively.

**As per claim 20**, refer to rationale as set forth in claims 12-13.

**As per claim 29**, this is the computer-medium product version of claim 1, hence is rejected with the corresponding rejection as set forth therein.

**As per claims 30-48**, these are the computer product claims corresponding to claims 2-20; hence are rejected with the corresponding rejections as set forth therein respectively.

6. Claims 21-28, and 49-56 are rejected under 35 U.S.C. 103(a) as being unpatentable over Elmroth et al., "A Web Computing Environment for the SLICOT Library", December 2000, Brite-Euram III, Networks Programme NICONET, in view of Research Systems, "IDL", copyright 1994, further in view of Shannon et al., "Mapping the Interface Description Language Type Model in C", November 1989, IEEE Transactions on Software Engineering.

**As per claim 21**, Elmroth discloses a method comprising:

providing a library file including functions defined by a first language (e.g. *SLICOT Library*, *BLAS*, *LAPACK* – pg. 1, Introduction; *Riccati equations* - Fig. 2-3; ...*uploaded ... Matlab files*, *data files*, *Latex*, *Scilab* – pg. 2, pg. 6, Fig. 1, 4 – Note: uploaded matrices in Matlab or Latex format , or Riccati equations read on library files - i.e. files served as input to a library generating tool - defined in one language, all such file integrated into the SLICOT library file framework);

processing the library file to create a function library (e.g. *SLICOT Routines* – Fig. 6) and a description file (e.g. PHP scripts – pg. 6-7), the function library including one or more functions defined by a second language, each function in the function library being translated version of a function in the library file (e.g. *SLICOT routines* – pg. 6; ...*written in C or Fortran* – pg.8: Conclusions and Future Work), and

using the description file to translate a program file from the first language into the second language (e.g. *SLICOT routines*, *Matlab binaries*, *PHP scripts* – pg. 7-8 – Note: library files xxxxMD or xxxOD files in conjunction with PHP scripts and converting math functions into m-files read on one or more functions translated from the library file in a second language, i.e. *Matlab binaries function files*).

---

But Elmroth does not explicitly disclose the description file including description information being sufficient to enable translation of a call to the function into a call to a corresponding function in a second language without requiring processing of the definition of the function; nor does Elmroth disclose that each call in the program file to a function in the library file is translated into a call to a corresponding function in the second language. Converting math

Art Unit: 2124

functions into another program like high-level programming language like Fortran or C (analogous to suggestion by Elmroth – pg. 8: Conclusions and Future Work) was a known concept at the time the invention was made. Research\_Systems, as set forth in claim 1, discloses definition language (IDL) and mapping various application functions to a fourth-generation programming language like C, and using such IDL file, i.e. description file similar to PHP scripts by Elmroth, to implement applications similar to the Riccati Equations or Matlab files by Elmroth, e.g. Math functions, graph plotting, Image Processing (see Research\_Systems, pg. 1-5). The high-level definition description file by Research\_Systems was an interactive definition language (IDL) well known at the time the invention was made for converting functions in one language into a corresponding functions in another language, like from Math functions to C program as taught by Research\_Systems. As set forth in claim 1, Shannon in a similar approach as Research\_Systems, also discloses mapping functions from one language to syntax and constructs implemented in C functions (re claim 1). Hence, it would have been obvious for one of ordinary skill in the art at the time the invention was made to provide an IDL as taught by Research\_Systems and Shannon to enhance the scripts by Elmroth so that the IDL description information is used instead of the PHP scripts, the IDL providing sufficient description information as to enable translation of a call to the function into a call to a corresponding function in a second language without requiring processing of the definition of the function; such that each call in the first language program file to a function in the library file is translated into a call to a corresponding function in the second language (re claim 1: Shannon). The benefits of using IDL would have been obvious because of the same reasons listed by Shannon: the IDL

Art Unit: 2124

provides type safe implementation into a high-level programming like C, and further does not require processing of the definition of the function ( see Shannon: Introduction, pg. 1333-1334).

**As per claim 22**, this claim includes the limitation as to translate a call to the function into a call to a corresponding function in a second language as in claim 21; hence is rejected as set forth therein. Further, Elmroth discloses a translated version of each function in the library file (*SLICOT Library, BLAS, LAPACK* – pg. 1, Introduction; *Riccati equations* - Fig. 2-3; *...uploaded ... Matlab files, data files, Latex, Scilab* – pg. 2, pg. 6, Fig. 1, 4- Note: uploaded matrices in Matlab or Latex format , or Riccati equations read on functions in the library file - i.e. files served as input to a library generating tool - defined in one language, all such file integrated into the SLICOT library file framework).

**As per claim 23**, this claim limitations correspond to those of claim 3; hence are rejected as have been addressed in claim 3, using most of Shannon teachings, mapping of function using a description file derived from examining a function in the first language library file, in light of Research\_Systems.

**As per claims 24 and 25**, these claims correspond to limitations of claim 3, 4 and 11; hence are rejected using the corresponding Shannon's teachings in view of the rationale to combine Elmroth, Research\_Systems and Shannon as set forth above.

**As per claim 26-28**, Elmroth discloses a Web call mapping and converting interface (Fig. 6), while Shannon discloses an interface to check call for error and type violation (the User Interface – pg. 1344). In light of the rationale as to combine the IDL teachings by Shannon with Elmroth's web interface and PHP scripts, the motivation to provide Elmroth's Web interface a function evaluation interface as suggested by Shannon would have been for the same reasons as

Art Unit: 2124

combining Elmroth with Shannon as in claim 21. Further, Elmroth does not specify variable input descriptor, variable output descriptor, descriptor for a known number of input or output arguments as recited in claims 18-20. In view of the rationale to combine Elmroth with the IDL by Research\_Systems and Shannon, these claims will be rejected as in claims 18-20 respectively.

**As per claim 49**, this is the computer-medium product version of claim 21, hence is rejected with the corresponding rejection as set forth therein.

**As per claims 50-56**, these are the computer product claims corresponding to claims 22-28; hence are rejected with the corresponding rejections as set forth therein respectively.

#### ***Response to Arguments***

7. Applicant's arguments filed 11/03/2004 have been fully considered but they are not persuasive. Following are the Examiner's corresponding counterpoints.

#### **As per the provisional obviousness-type double-patenting rejection:**

(A) In reply to arguments (Appl. Rmrks, pg. 3), the rejection has pointed out what portion(s) of the instant claim, albeit covering less details, has been anticipated by the portions of the '288 claim. Precisely, the rejection has mapped what in '288 claim 16 reads on a 'definition of a function', a 'first language', 'description information about the function', 'translation of a call to the function into a corresponding function', 'in a second language' and 'without requiring processing of the definition of the function' of the instant claim. Even though the instant claim appears less elaborate in details than the combined details of copending claim 16, the subject matter claimed in the instant claim has been anticipated by, or amounts to an obvious broader variation of the combined details of '288 claim 16 including inherent or equivalents derived therefrom; and this is a proper provisional obviousness-type double patenting rejection.

**Rejections under 35 USC §103(a):**

(B) As for claims 1 and 29, Applicants have submitted that Shannon does not write in a source language; but instead uses data declarations and utilities representative of the target language; hence does not teach or suggest processing a function written in a source code to create description information about that function ( Appl. Rmrks, pg. 5, middle). This is not persuasive. First, there is reciting of a 'source language'. Second, there would be no need to make use of an IDL if there is no heterogeneous programming language environments so to implicate a first language and a target language because a second or target language by itself would not require a tool to provide conversion definition and mapping as taught by Shannon, i.e. a otherwise regular compiler using data structures in a symbol table would suffice to link/generate a target program. For one skill in the art, IDL is a intermediate form of language being used for mapping constructs of a targeted code language; and IDL starts from a base conceptual constructs of existing procedural programming language. Such language is mostly written in a different language than the target language but able to construe functionalities of a useful domain or application of interest that would otherwise be expressed in different format -- as shown by Research\_Systems or MathLabs ( see Application Background) or even UML/COM modeling. This amounts to what Shannon was alluding to when Shannon mentions about

---

starting with structures by *existing languages* and use IDL to target languages as ada or ABC – via Diana or Plum - C compiler and pertinent C constructs ( see pg. 1333 bottom right column to pg. 1334, left column; Specifying Data structures in IDL – pg. 1334, right column). The notion of having a descriptive intermediate conversion tool based on existing programming language structures is therefore seen as fulfilling the claim limitation as to 'definition of a function



Art Unit: 2124

associated with a first language to create a description information about the function', when the claim is interpreted as reasonably and broadly as allowed by one of ordinary skill in the art.

Besides, the claim does not provide more specifics as to what *function* being associated with a *first language* consists of so as to make it undeniably different from the above interpretation.

Further, the functions being analyzed in a first language so that the intended functions are being analyzed/created by Shannon via IDL class declaration or IDL specifications (or by

Research\_Systems for the math equations) are interpreted as fulfilling the concept of functions being processed and subsequently served as input into the IDL translator.

(C) Applicants have submitted that Research\_Systems fails to teach or suggest processing a function written in a source language to create description information ( Appl. Rmrks, pg. 5, bottom- pg. 6, top). It is noted that the IDL tool is to provide all the necessary description information enabling a target language to be converted therefrom; and by analyzing the Math functions relationships as mentioned in the rejection, Reseach\_Systems has fulfilled the processing of functions as recited. Because the math functions imply some math constructs presented in some original format, such format read on a first language; and Reseach\_Systems has taught the existence of such math functions from which a need to convert incurs. The combination as set forth has pointed out why the combination adding Research\_Systems to the IDL tool of Shannon would be obvious; and Applicants fail to show in what specific way such combination would be inappropriate.

(D) Applicants submitted that Shannon does not teach information derived from processing code written in a first source language (Appl. Rmrks, pg. 6, middle – Note: 'source language' is not claimed). Section B has shown that Shannon has description information derived from

Art Unit: 2124

constructs known from existing procedural languages and Research\_Systems has disclosed that some Math functions being used for some applications can be further implemented into target code via Shannon's IDL. So long as Applicants fail to point out why the combination from the rejection is unfounded or yield adverse effects; then section C above suffices to address the instant argument put forth in this section.

(E) As to arguments on claims 3, 23-25, 31, and 51-53, Applicants have submitted that there is no teaching or suggestion on 'taking a function already written in one language, creating description information ... into target language' ( Appl. Rmrks, pg. 6, bottom, pg. 7, top). As pointed out in section B, analyzing a functionality of a domain of interest, like a mathematical equation in a math format and generate constructs in IDL is viewed as fulfilling what Applicants call 'creation of description information about a function in a first language ( Note: the claim does not recite 'written in source programming language'). The recited concept of 'creation of description information' is too broad to be interpreted other than the cited portions by Shannon or Research\_Systems and what these references try to accomplish when a conversion tool like IDL is transposing functionalities of an application into target constructs.

(F) Applicants have submitted that library in C is not 'translated function into a library of entries'; and that the references fail to teach 'parameter in a source language into a description information' (Appl. Rmrks, pg. 7, 3<sup>rd</sup>, 4<sup>th</sup> para). There is no reciting of 'source language'; and besides, alleging that the references do not have what is claimed is not necessarily convincing and specifically pointing out what distinguishes the claim from the prior art. The referred to parts of Shannon will stand as anticipating the limitations in view of the lack of specifics in the claim and the broad reasonable interpretation that have been applied when construing the

Art Unit: 2124

limitations as superficially recited. Library entries and parameters are always entered in any compilation scheme let alone the IDL compiler as disclosed by Shannon while the concept of having language constructs originated in a different language has been addressed in section B above.

(G) Applicants have submitted that analyzing input/output declaration in C is not 'creation of another parameter in a target language'; and that the references fail to teach 'translating of parameter in a source language into a description information ... used in a later translation into a target language' (Appl. Rmrks, pg. 7, bottom para). The recited limitation as to 'description information' is too broad to distinguish from the reference, e.g. the IDL specifications as metadata shown in Fig. 3 as input to the translator by Shannon is fulfilling *description information*. The concept of parameters being declared and translated in a compiler is inherent in any such process wherein a parameter from an input to a translator is converted in another form of parameter at the output of a translation.

(H) Applicants have submitted that 'extraction of scope and description of the scope for use in later translation into a target language' (Appl. Rmrks, pg. 8, 2<sup>nd</sup> para ). The rationale here falls under the ambit of the how the arguments have been addressed in section G; hence the arguments herein are referred thereto.

---

(I) Applicants have submitted that the references fail to suggest or teach 'processing of a definition of a function ... information' (Appl. Rmrks, pg. 9, top para). This is referred back to the corresponding response as set forth above.

(J) Applicants have submitted that the 'claimed invention processes a function ... source language ... into a target language' (Appl. Rmrks, pg. 9, 3<sup>rd</sup> para ) and that the combined

Art Unit: 2124

teachings of Shannon and Research\_Systems fail to disclose 'the integral step ... a source language ... to translate a source language ... target language' (Appl. Rmrks, pg. 9, bottom to pg. 10, top). There is no recitation of a 'source language' in the claims and the processing of a function limitation has been addressed above.

(K) Applicants have submitted that there is no implicit teaching of 'retrieval of an item from a file of description items' (Appl. Rmrks, pg. 10, 3<sup>rd</sup> para). The basis of anticipation (or anticipation by obviousness) is not based on explicit recital of features but also on implicit features taking into consideration the level of prior art understood at the time of the invention combined with well-known concepts construed by one ordinary skill in the art and *inter alia*, the scope of what is claimed. The fact that the rejection expresses that Shannon does not explicitly disclose some claimed elements does not translate into the admission that the Shannon does not disclose at all such elements in view of the above rationale. And Shannon's retrieval of declared constructs in a IDL specifications encompasses the above limitation about retrieval of an item because there would be no specifications without retrieval of entries thereof. Besides, the recited concept of 'creation of description information' is too broad to be interpreted other than what Shannon or Research\_Systems try to accomplish when a conversion tool like IDL is transposing functionalities of an application into target constructs, such transposing retrieving data declared in the IDL specifications.

(L) Applicants have submitted that neither Shannon nor Research Systems ... Examiner admits that Elmroth would not teach ... claimed invention' and that the references fail to teach or suggest 'generating a call ... definition of a function' (Appl. Rmrks, pg. 11, 1<sup>st</sup> and 2<sup>nd</sup> para ).

Art Unit: 2124

These arguments are referred back to section K above and the observations pointing out how broadly the 'description information' limitation amounts to in previous sections.

For the above observations, the rejection will stand as set forth above.

***Conclusion***

8. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (272) 272-3735. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

---

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on (571)272-3719.


The fax phone number for the organization where this application or proceeding is assigned is (571) 273-3735 ( for non-official correspondence – please consult Examiner before

Art Unit: 2124

using) or 703-872-9306 ( for official correspondence) or redirected to customer service at 571-272-3609.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

VAT  
March 10, 2005

  
**KAKALI CHAKI**  
**SUPERVISORY PATENT EXAMINER**  
**TECHNOLOGY CENTER 2100**